Approaching the Physical Limits of Computing

Michael P. Frank
FAMU-FSU College of Engineering
Department of Electrical & Computer Engineering
mpf@eng.fsu.edu

Abstract

As logic device sizes shrink towards the nanometer scale, a number of important physical limits threaten to soon halt further improvements in computer performance per unit cost. However, the near-term limits are not truly fundamental, and may be avoided by making radical changes to the physical and logical architecture of computers. In particular, certain assumed limits to the energy efficiency of computers have never been rigorously proven, and may be circumvented using physical mechanisms that recover and reuse signal energies with efficiency approaching 100%. However, this concept, called reversible computing, imposes tight constraints on the design of the machine at all levels from physics to algorithms. We review the physical and architectural requirements that must be met if real machines are to break through the barriers preventing further progress, and approach the true fundamental physical limits to computing.

1. Introduction

Computer scientists, engineers and futurists alike enjoy citing "Moore's Law," the famous 1965 prediction by Intel co-founder Gordon Moore [1] that the number of devices per chip would double every year (later adjusted to every 18 months [2]), with accompanying benefits in performance per unit cost. Although the (revised) prediction has held true for 40 years, it is of course only a technological trend, not a law of physics; in any direct conflict between the two, we can be assured that physics will win.

In fact, it is by now fairly common knowledge among experts in the semiconductor industry (*e.g.*, see [3]) that raw chip performance (if not bit density) must level off within just a few years, as conventional transistor circuits approach limits to their performance at reasonable levels of power consumption, limits that ultimately arise from quite fundamental laws of statistical mechanics and thermodynamics. However, it is also true (but much less widely recognized) that these limits have never been validly proven to apply to *all* physically possible computing technologies. Rather, they are apparently just a consequence of certain contingent historical attributes of the presently-dominant paradigm for digital logic, namely its reliance on the use of *irreversible* (that is, many-to-one) operations as the basic primitive logical events in computation, which in turn requires using physical mechanisms for logic that are also *thermodynamically* irreversible [4]; in present technology, logic signal energies are allowed to dissipate themselves (to the form of heat) whenever those signals are manipulated.

But in fact, the thermodynamic irreversibility of logic can be avoided, but *only* if we abandon logical irreversibility and instead adopt reversible (one-to-one) operations as our primitive events in computing, a fact that was first recognized in 1973 by the famous IBM researcher Charles Bennett [5], who, incidentally, later co-invented both quantum cryptography and quantum teleportation, and who remains today an active and widely-respected leader in the growing fields of quantum information and quantum communication.

Although the theoretical concept of reversible computing has been known (at least to a small community) for the last 32 years, and much progress has been made in the theory and engineering of reversible machines in the meantime, the new paradigm has not yet achieved a level of practicality that would foster extensive commercial applications. This is due primarily to two reasons: (1) Developing a cost-effective technology for reversible computing that actually saves significant energy in practice is a far more complex and difficult engineering problem than most computer science theorists (even those within the field) tend to imagine, and (2) during most of this period, the limits to the energy efficiency of conventional technology were still very far away from being reached, and so, further refinement of the conventional technology was a much easier and less disruptive path for technological evolution to follow. However, reason #2 is on the verge of going away, now that the semiconductor industry is running out of ideas for pushing the conventional technology much further,

Proceedings of the 35th International Symposium on Multiple-Valued Logic (ISMVL'05)
0195-623X/05 $ 20.00 **IEEE**

and so, today the time is ripe for increasing numbers of physicists, engineers, and computer scientists to turn increased attention towards solving the difficult problems that today still prevent reversible computing from being a marketable solution to the imminent power-performance crisis.

In this paper, we review the present technological situation as it stands, with an emphasis on the physical and logical design requirements that must be satisfied if we wish to overcome the near-term limits to computer efficiency, which will require making reversible computing technology practical.

The structure of this paper is as follows: Section 2 reviews the thermodynamic foundations of the near-term limits on computer performance. Section 3 raises the possibility of surpassing these limits by improving computer energy efficiency. Section 4 discusses why reversible computing is an absolute physical prerequisite for accomplishing this to an extent that will get us very far beyond the present limits. Section 5 summarizes the detailed physical requirements for new reversible logic technologies to exceed conventional logic performance, and reviews a few of the present efforts to implement such technologies, and then reviews the logical, architectural, and algorithmic requirements for the design of efficient reversible systems. In the long run, as we approach ever closer to 100% energy efficiency, all of computer science will eventually need to change in order to accommodate the new reversible paradigm [6]. It is a great challenge, but also an exciting opportunity. Finally, section 6 discusses some long-term implications of this work, and concludes the paper.

2. Limits of conventional technology

Let us begin with a few basic definitions and symbols, to be sure that we are all speaking the same language. *Power* (P), in physics, refers to the amount of energy that undergoes some transformational process (*e.g.*, transmission between bodies or dissipation to heat) per unit time ($P = E/t$), while *performance* (G) in computing refers to the number of standardized computational operations that are dynamically performed per unit time ($G = N_{ops}/t$). *Power-performance* (G_P) in computer engineering [7] simply means performance per unit power dissipated, $G_P = G/P_{diss}$, which (as a direct result of the previous definitions) is also equal to N_{ops}/E_{diss}, the number of operations performed per unit of available energy that gets dissipated to heat. In realistic scenarios, the dissipation of a unit of energy to heat typically imposes non-negligible direct and indirect economic costs, and so, power-performance often sets a lower bound on the cost of performing a computation having a given complexity N_{ops}, or in other words, an upper bound on the complexity of a computation that can be performed within a given energy budget. Thus, to improve our practically- affordable computational capabilities, we must continue to improve power-performance, or in other words, reduce the energy dissipated per operation.

How much energy must be dissipated in order to perform a computational operation? Let us first consider a very simple operation, such as changing a bit from a 1 to a 0. If the 1 is represented by a physical entity or *signal* (*e.g.*, a high voltage on a wire) that carries an amount of available energy E_{sig}, then one simple way to clear the bit would be to allow this signal energy E_{sig} to simply dissipate away into the environment the form of heat. This is in fact exactly what happens in digital circuits today: a 1 is cleared by connecting a wire to ground, and allowing its stored electrostatic energy to dissipate away. (With this particular mechanism, we see that $E_{diss} = E_{sig}$; later, we will see that in other mechanisms, E_{diss} can be made much less than E_{sig}.)

Now, what are the lower bounds on E_{sig}? These follow from reliability requirements and thermodynamics. One of the most elementary textbook facts about thermodynamics is that a small subsystem of a system at temperature T has a probability proportional to $e^{-|\Delta E|/kT}$ of being found to be in an accessible state having a free energy that is ΔE away from the subsystem's expected (average) free energy, as a result of thermal fluctuations. In this formula, $k = 1.38\times10^{-23}$ J/K is Boltzmann's constant, which is just the log e unit of entropy. This result is a classical one, but the quantum corrections to it are small when $\Delta E >> kT$.

Thus, if we want our signal, when measured, to have a probability $p_{err} << 1$ of being found to be in an incorrect state, and probability $1 - p_{err} \approx 1$ of being found in the correct state, then there must be a difference of $|\Delta E| \geq (kT \ln r)$, where $r = p_{err}^{-1}$ is the reliability factor, between the free energies of the correct and incorrect states. Therefore, logic signals that have only probability p_{err} of being invalid must involve this much energy, in the sense that energy transfers of this magnitude are required to transmit and transform those signals. As an example, if we wish a signal to have an error probability of only $p_{err} = 10^{-40}$, then the signal needs to involve energies of at least $92.1\ kT = 2.38$ eV at room temperature.

One may at first wonder whether the energy requirements of signals could be lowered arbitrarily by just decreasing the internal temperature T_{int} of the system. Unfortunately, it turns out that this

strategy actually cannot help to reduce total system energy dissipation, because the effective T when considering *total* system energy dissipation is always that of the outside environment to which waste heat is being released. This is because the internal dissipation of energy ΔE results in an entropy increase of $\Delta S = \Delta E/T_{\text{int}} = (kT_{\text{int}} \ln r)/T_{\text{int}} = (k \ln r)$ which (note) is *not* temperature-dependent. This entropy cannot be destroyed (by the 2nd law of thermodynamics), and so it must ultimately be expelled into the outside environment at temperature T_{env}, resulting in a total energy dissipation to the environment of $E_{\text{diss}} = T_{\text{env}}\Delta S = kT_{\text{env}} \ln r$. As long as we must expel our waste heat to the atmosphere, we are stuck with $T_{\text{env}} \approx 300$ K.

The use of error correcting codes does not really help either, because the fundamental relationship between error probability and bit energy is derived in a way that does not depend on *how* the bit is encoded. A bit that is encoded using some fancy redundant coding scheme that permits error correction would require exactly the same (if not more) total energy to achieve a given level of reliability as would an isolated bit. Thus, there is nothing that can be done to improve reliability at higher levels in the system design that is any more energy-efficient than simply "bulking up" the energy content of individual bits. For this reason, I don't believe that the energy per bit in practice will ever shrink much below a room-temperature equivalent energy on the order of 1 eV, since below that level, systems become too unreliable ($p_{\text{err}} > 1.6\times10^{-17}$) to carry out large-scale computations without error.

Even worse, in practice, this minimum energy applies not only to entire logic bits, but also to the individual particles (such as electrons) that make up the signal. Logic signals today are typically encoded using relatively large numbers of electrons (~10^4 in the smallest logic nodes in the latest technologies), and all of these electrons must be presented with energy barriers of this magnitude in order for them not to experience "error" (*e.g.*, by jumping across the channel of a turned-off transistor) and rapidly leak away, erasing the stored bit and dissipating its energy. Thus, total logic signal energies today are ~10^4 eV, or on the order of 1 fJ, which translates to a power-performance level of only about a million logic operations per nanosecond, per watt of power consumed. A million logic operations within a GHz clock cycle may at first sound large, but it is only sufficient for on the order of perhaps 10 typical double-precision floating-point operations, so this leads to a maximum performance per watt for today's technology of only around 10 GFLOPS, even in an ideal special-purpose architecture that imposed no further overheads beyond those of the arithmetic.

Further performance gains within the conventional logic paradigm will thus require further reducing the number of electrons per bit, which may yet happen—researchers all over the world are experimenting today with techniques for manipulating even single electrons, and using them in switching. However, ordinary field-effect transistor technology does not appear capable of being scaled to work with much less than ~100 electrons (if that), which only gets us a factor of ~100 beyond today's power-performance levels, or a mere ~10 years (until 2015) on the historical performance trendlines.

Even if the number of electrons per bit can be reduced all the way down to 1 by (most likely) abandoning conventional MOSFET technology, if the electrons' energy is still thrown away with every operation, as is done today, that still only gets us another factor of 100, or to the year 2025.

It is important to note that the argument for the energy limit at this point does not in any way depend on the physical medium of the technology, for example, as to whether it uses electrons or photons or molecules to store information. The laws of thermodynamics are universal, and so any information storage medium must involve signal energies (or energy barriers) of magnitude 40 $kT \approx 1$ eV $\approx kT \ln 10^{17}$ in order to function with a reasonable level of reliability, and thus, energies of such magnitudes must be manipulated whenever those signals are processed (moved or modified). If we continue to insist on discarding the entire signal energy and dissipating it to heat every time we manipulate a signal, then raw (gate-level) computer performance is doomed to reach a plateau that is (at most) 20 years in the future, at historical rates.

3. The need for energy efficiency

The one (and only) loop-hole in the above pessimistic scenario is that it depended on the assumption that energy of the same magnitude as the entire signal energy gets dissipated to heat whenever a logic signal is manipulated, or in other words that $E_{\text{diss}} \approx E_{\text{sig}}$. However, there is no fundamental reason why this must necessarily be so. Suppose instead that $E_{\text{diss}} < E_{\text{sig}}$. In other words, in performing a given digital operation, suppose that a fraction $f = 1 - E_{\text{diss}}/E_{\text{sig}}$ of the signal energy (where $0 \le f \le 1$) is retained in an organized form that can reused in performing subsequent operations. We then refer to the fraction f of energy that is recovered for later use as the *energy efficiency* of the operation. (Like the energy efficiency of a transformer, it can be expressed as a percentage be-

tween 0% and 100%.) Present-day computation has an energy efficiency of basically 0%, in other words, essentially *all* of the signal energy is dissipated to heat in each operation that changes the logic value of the signal. However, perhaps surprisingly, this is not a necessary feature of computation; in fact, there is no proven absolute upper limit on the energy efficiency of computation that is less than 100%.

As a concrete example, to clear a voltage-coded bit with arbitrarily low dissipation, one can connect it to a changing reference signal that is initially at the same voltage, and that goes to the ground level at a steady rate over time t. The power dissipation during this process is $P = IV$ where $I = Q/t$ is the current and $V = IR$ is the voltage drop along the discharge path. The energy dissipated is $E = Pt = IVt = I^2Rt = Q^2R/t$, which becomes arbitrarily small (for a fixed charge Q and resistance R) as the charging time t is made longer, with no lower limit. This is an example of *adiabatic switching*, which is the basis for many approaches to energy recovery.

Although there is no general lower limit on energy efficiency, if we are given a fixed level of reliability r in the logic, the energy efficiency of computation is at most $f < (1 - r^{-1}) = (1 - p_{\text{err}}) = p_{\text{ok}}$ (the probability that the bit value is correct), since the occurrence of an error in a given bit essentially represents the dissipation of that bit's energy away from the intended computational trajectory. However, recall that reliability itself can be increased exponentially by simply increasing the bit energy, since $r = p_{\text{err}}^{-1} = \exp[E_{\text{sig}}/kT]$ for errors due to thermal noise. Increasing reliability in this fashion increases signal energy, but only logarithmically in reliability, and so it *reduces* the reliability-related lower limit on *overall* energy dissipation, since $E_{\text{diss}} = E_{\text{sig}}(1 - f) > E_{\text{sig}}r^{-1} = E_{\text{sig}}p_{\text{err}} = (kT \ln r)\exp[-E_{\text{sig}}/kT] = (kT \ln r)/r$, which approaches 0 as $r \to \infty$. So, we can have *both* arbitrarily high reliability *and* arbitrarily low energy dissipation per op in the face of thermal noise at a fixed temperature, as long as we make the bit energy as large as necessary for this, and no *other* factors prevent the reliability r from approaching ∞, or the efficiency f from approaching 1, or the dissipation E_{diss} per operation from approaching 0.

There is good reason to think that f can indeed approach 1 if we consider the "manipulation of information" to just be a way of interpreting a more general class of physical processes, which is the transformation of a physical system from one state to a different (distinguishable) one. A wide variety of physical systems involve the transformation of the physical state from one form to another distinct one with extremely high efficiency, that is, with an extremely low fraction of the system's energy being dissipated per transformation step. In cyclic processes, the energy efficiency is commonly characterized by the quality factor $Q = f / (1 - f)$. Many examples are known of physical systems (both macro-scale and nano-scale) having Q factors in the billions or higher, corresponding to an energy efficiency of ≥99.9999999%. For example, a crystal or molecule vibrating in a vacuum, a planet orbiting a star, and a precessing quantum spin are all examples of systems that are known to carry out large numbers of transitions between distinguishable states with extremely high energy efficiencies.

Also, in quantum mechanics, if we know the laws of physics with high precision, and have prepared an initial quantum state also very precisely, and have well isolated it from undesired interactions with its environment, the rate of entropy increase in that quantum system as it evolves can theoretically be arbitrarily close to zero, corresponding to arbitrarily little of the system's energy being dissipated away from the predicted trajectory of the system. This is because the evolution of the quantum state (wavefunction) of an isolated quantum system is (contrary to widespread misunderstandings) fully deterministic and non-chaotic (in fact, it is linear); the apparent nondeterminism of quantum mechanics only arises (emergently) when a system interacts with and leaks information to an uncontrolled outside environment.

As time goes by, and we characterize the fundamental constants of physics ever more precisely, and we learn how to manufacture and simulate quantum devices with ever more accuracy, there is no reason to think we cannot get to the point where we can eventually design and construct nanoscale quantum systems that glide along complex trajectories that pass rapidly through large numbers of distinct quantum states with negligible entropy increase. It is "simply" a matter of sufficiently accurately characterizing and modeling the system's quantum dynamics.

Once we have attained this core capability of high-precision, nanoscale quantum engineering, it is then "simply" a matter of designing and building particular systems whose natural (and entirely predictable) dynamics takes them through a sequence of quantum states that corresponds exactly to a preprogrammed sequence of computational states, in other words, to a desired computation. There is no reason from fundamental physics to think that this capability cannot eventually be possible (and even quite practical) to achieve. However, realizing it imposes stringent constraints on the design of our lowest-level physical devices and mechanisms for performing digital state manipulations, and on the

logical and computational architecture of highly energy-efficient computers based on such devices.

Energy efficiency well above 0% is already empirically known to be possible, in fact, there already exists today in the chip-design community a burgeoning community of engineers who are exploring energy-recovering logic techniques in depth. In principle, these are (very roughly) analogous to the regenerative braking systems on modern hybrid cars, which recover some of your car's kinetic energy when you step on the brake, and return it to the battery, rather than the conventional approach of just dissipating it as heat in the brake shoes. In a somewhat similar way, a good circuit design can arrange for most of the energy stored in a logic signal to be returned to the power supply when that signal is erased, rather than being dissipated to heat along some resistive discharge path. However, the maximum energy efficiency of most standard approaches to energy recovery is rather limited, and the usual techniques for this do not actually get extremely close to 100% efficiency.

4. Requirements for sub-*kT* computing

We now ask, what is required in order for computers to attain arbitrarily high levels of performance at fixed levels of power consumption? That is, what is required in order for the energy efficiency of computation to get arbitrarily close to 100%, so that the energy dissipation of individual logic operations can plunge from the >40 kT level of conventional logic, to arbitrarily lower levels, even to levels much less than kT? Quite rigorously, we can show that what is absolutely necessary for this is *reversible logic*. (This was already known to the late Rolf Landauer of IBM as early as 1961 [4], although at the time, he did not yet realize that a reversible solution was logically possible.)

To understand the connection between thermodynamic reversibility (which means high energy efficiency, low energy dissipation, low entropy increase) and logical reversibility (which means the use of 1-to-1 transition functions for computing), we first need to step back and review some more basic facts of physics.

We mentioned earlier that quantum dynamics, when modeled precisely, is completely deterministic. This is well-known among top quantum physicists; indeed, it is etched into the very mathematical core of quantum mechanics. The apparent non-determinism of quantum phenomena is perfectly validly (and, arguably, by far most simply) explained and interpreted as merely the expected appearance that arises whenever a quantum system interacts with and becomes "entangled" (correlated in a quantum way) with any complex external environment that won't soon happen to act in a way that would disentangle the state of the environment from that of the system. The apparent non-determinism of quantum events only arises because the well-determined quantum correlations that persist in the resulting state can't be seen by local observers, who can only interact with the small fragment of the quantum state that immediately surrounds them in configuration space; this fragment appears random to them, simply because it is an arbitrary piece of a full ensemble. Nevertheless, the global state evolution always remains fully deterministic, and in principle, completely predictable.

All of our most successful physical theories encompass and reflect this underlying determinism, including quantum electrodynamics, which predicts details of atomic spectra to 9 digits of precision, and general relativity, which predicts gravitional energy emission rates in binary pulsars with comparable accuracy. In essence, *all* of the vast, overwhelming mountains of empirical evidence collected by experimental physicists has so far proved to be perfectly consistent with this modern understanding of quantum physics. It is, mathematically and conceptually, the simplest comprehensive and accurate theory that we have, so Occam's razor ought to convince all good scientists that we should accept it at face value [8].

Now, a key element of the mathematical structure of all modern physical theories is the principle of Hamiltonian dynamics, which says that the state vector x evolves over time according to a differential equation that is first order in time, $dx/dt = g(x)$, where g is a function of the instantaneous state x. The very form of this differential equation makes the state evolution deterministic. But it is crucial to note that it is also *reverse-deterministic*, that is, deterministic in the negative time direction; this is simply because dt in the Hamilton's equation can be equally well either positive or negative.

Consider now the transition function $F_{t,u}(x)$ that maps old states to the resulting new states between two times t and $u > t$, that is, $x(u) = F_{t,u}[x(t)]$. Because the dynamics is reverse-deterministic, the transition function $F_{t,u}$ must be a one-to-one (injective) function over the state space $X = \{x\}$. To say that the transition function is injective is the very definition of a *time-reversible* (or just "reversible" for short) dynamics. Since all successful theories of fundamental physics admit a Hamiltonian formulation, all of them share this core property of dynamical reversibility.

Reversibility is thus one of the most fundamental, unavoidable, and universal characteristics of modern physics. Without it, terrible things would

happen: the second law of thermodynamics would not always hold, probabilities would not always sum to 1, energy would not be conserved; *etc.* Even the famous astrophysicist Stephen Hawking recently conceded a bet he had made, and admitted that even objects as extreme as black holes still develop in an entirely time-reversible fashion. So, we can be very confident that, no matter what new developments may occur in theoretical physics in the future, dynamical reversibility will always remain a fundamental cornerstone of physics.

Now, we will show why dynamical reversibility immediately implies that logical reversibility is a requirement for thermodynamic reversibility. We will do this by showing that logical *ir*reversibility implies thermodynamic irreversibility.

Suppose a given computational operation that we perform within a machine is logically irreversible. This means that the logical transition that occurs within the machine is described by some logical transition function L that takes "before" (predecessor) states to "after" (successor) states and which is not a one-to-one function. For example, the operation of unconditionally clearing a bit that could have been either 0 or 1 beforehand means that $L(0) = 0$ and $L(1) = 0$. Since two different operand values (0 and 1) produce the same result (0), this transition function L is not one-to-one, and so the operation that it carries out is by definition not logically reversible.

Given that physical states always develop in a one-to-one fashion, how can the bit-operation "clear" be performed? Only by embedding this operation within some larger physical process in which the "erased" information is preserved in some form. For example, suppose we conjoin our bit (0 or 1) with some other system (the "environment") whose initial state is S. The combined states of the bit and the environment can then be labeled 0S and 1S. Now we wish to clear the bit. We can do this by taking 0S→0S and 1S→0T, where T is some other state of the environment. Now the bit is unconditionally 0, but the environment has gone from being in a known state (S) to being in an unknown state (either S or T).

In general, given the dynamic reversibility of the laws that govern our physical universe, *information can never really be destroyed*, it can only be moved from one system (in this case, the 0 or 1 bit) to another (in this case, the environment).

If the system that we're calling the environment really is an uncontrolled physical environment (for example, if we're releasing the discarded information in a flow of heat into the atmosphere), then we can really never get the discarded information back, because we can't expect the outside environment to be so kind as to return the information to us in pristine, undisturbed condition. Thus, the known information that was originally contained in the bit to be erased has been degraded to the form of *unknown* information (*i.e.*, entropy) in the environment. Total entropy has thus increased, by the amount of 1 bit, and entropy increase is the very definition of thermodynamic irreversibility.

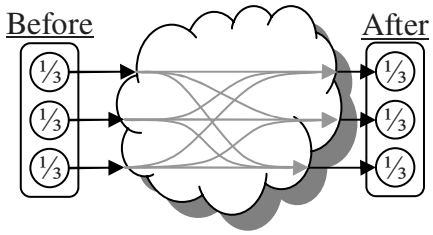

Figure 1. Only nondeterministic processes such as this one can be both non-injective and thermodynamically reversible.

If the environment is at temperature T, then this means (by the very thermodynamic definition of temperature, $1/T = \partial S/\partial E$) that increasing its entropy by a small amount dS will require dissipating an amount $dE = T\,dS$ of energy to the form of heat in that environment. To increase the entropy of our environment by an amount 1 bit = log 2 = (log e)(ln 2) = k ln 2 thus requires that we dissipate at least $E_{diss} = kT$ ln 2 energy into that environment. This implies that the energy efficiency of a bit-erasure operation, when the bit is lost, cannot be greater than $f = 1 - (kT \ln 2)/E_{sig}$. For example, bit-erasure efficiency can be at most 98% for signal energies of ~40 kT that are sufficient to ensure reliability in the face of thermal noise. (However, achieving even this high a level of energy efficiency will require aggressive energy recovery, well beyond what is already being done today.)

Now, however, if the "environment" is instead another controlled component of our system, then there is no implication that entropy is increased in the "bit erasure" (really, bit movement) process. In this case, the process can be thermodynamically reversible. But, a controlled component that can store a bit can also be considered to be part of the *logical* state, and the joint logical state is then updated injectively. Therefore, the physically reversible version of the bit movement process can also be considered to be logically reversible.

One minor caveat to this discussion is that the transition relation in a thermodynamically reversible process may actually (more generally) be N-to-N, not necessarily just 1-to-1. For example, see Figure 1 above. On the left is a group of three equally likely initial logical states, each with probability 1/3. This distribution has an entropy of log

3, or about 1.1 k. The cloud represents a physical process whose details are unknown, either because the physical laws are uncertain, or because the system is interacting with an unknown environment. On the right are three possible final logical states that the system can end up in. Thus, the logical dynamics here is not modeled as a one-to-one function, but rather as a "3-to-3" nondeterministic relation in which probabilities may be attached to the transitions (arrows) between before and after states. The equilibrium distribution over initial states will be a stationary point of the dynamics as long as the transition probabilities satisfy the property of *semi-detailed balance*, which says that $\sum_b p(b \to a) = 1$, where a is a general after-state, b ranges over before states, and $p(b \to a)$ is the probability that before-state b will transition to after-state a. In this case, the after-states will remain equiprobable, and the entropy will remain constant. So, the process described need not generate entropy and thus may be thermodynamically reversible, even though the logical operation performed is not an injective one. However, notice that if the number of possible after states was *less* than the number of before states, then the entropy in the logical state could *not* remain the same, and so in that case, logical state information would have to be expelled into the environment, where it would become physical entropy and imply energy dissipation into the environment.

Thus, the true logical requirement for thermodynamically reversible computing is not really that the logical operation performed must carry out a deterministic 1-to-1 transformation of the logical state, but only the somewhat weaker requirement that the operation performed must not *decrease* the number of possible logical states, while the transition function must have semi-detailed balance. Thus, the operation performed may in fact be, strictly speaking, *logically* irreversible, as long as it is *also* nondeterministic (in the sense of a probabilistic or randomized computation). However, if it is deterministic, then it must also be injective.

Also, in order for nondeterministic, non-injective N-to-N operations to be thermodynamically reversible, the initial state must be truly random (unknown) already—if not, then its subsequent randomization by this operation is a form of entropy increase, and is itself thermodynamically irreversible. But if the state is already randomized, and it is only being re-randomized by the N-to-N operation, then it is unclear what the computational utility of such an operation would be.

Incidentally, note that nondeterministic 1-to-N operations (or more generally, N-to-M operations, where $N < M$) actually *increase* the entropy of the logical state, and thus, in principle, can be used to (temporarily) reduce the entropy of the environment, and gain energy from it. (Essentially, the part of the logical state that is being randomized can be used as the cold reservoir in a heat engine.) Later, an M-to-N operation can be used to reduce the number of logical states again, and at this time we will have to pay back the energy we have borrowed from the environment. As long as the overall process is still N-to-N, it may still be thermodynamically reversible. As a result, reversible computations may freely utilize randomized (probabilistic) algorithms without paying any extra thermodynamic penalty for doing so.

Although randomized algorithms are useful for many purposes, for simplicity we will focus on the use of deterministic, 1-to-1 operations below.

A second, more important caveat to the above discussion is that thermodynamic reversibility does not actually require the logic operation to be 1-to-1 over the set of *all* conceivable input states, but only over the restricted subset of those input states that can actually arise, given the architecture of the machine. We will say that such an operation is *conditionally reversibile*, since it is reversible under the condition that certain restrictions on its inputs are satisfied. The concept of conditionally reversible logic operations is a very useful one, because conditionally reversible logic operations often turn out to be easier to implement than fully reversible ones. We will see examples in section 5.6 below.

Finally, although ordinary deterministic N-to-1 operations (such as bit erasure) are thermodynamically irreversible, we can in fact emulate ordinary computations that are composed of such operations by embedding them within equivalent computations that are composed entirely of 1-to-1 operations. This was the key insight that was developed by Landauer and Bennett, and it enables reversible computers to remain computationally universal (Turing-complete). We will discuss how this embedding works in section 5.7.

5. Requirements for reversible logic

The previous section showed why achieving significant performance improvements well beyond the limits of conventional technology requires that we move to a new computing paradigm that is primarily based on logic operations that carry out 1-to-1 transformations of the computational state. (The only exception is for many-to-one operations that are only used to erase random bits that were previously obtained from the environment.)

In this section, we review the key engineering requirements that are imposed on computer designs

by the need to use reversible logic. These requirements constrain the system design at every level, from the modeling of physical processes used to implement logic operations to (eventually) the design of high-level software algorithms. We divide these new requirements into constraints on: (1) device physics modeling, (2) energy recovery mechanisms, (3) logical state encodings, (4) logical transition processes, (5) synchronization mechanisms, (6) logic gates, (7) functional unit designs, (8) processor architectures, (9) hardware design languages and software programming languages, and (10) application algorithms. As we go through these, we mention the progress that has been made to date towards meeting these requirements.

5.1. Device physics modeling

In order to create reversible devices that dissipate much less than kT energy per operation, nanoscience must progress to the point that we are able with high accuracy, to track what happens to the "cloud" of probable physical states representing a given logical state over the course of a given storage, communication, or logical transition event. This breaks down into several requirements.

Thermodynamic reversibility requires that our uncertainty about the state must not increase much over time, that is, the "cloud" of probable quantum states of the system must not spread out very much over the course of the event. For example, in the simple case of flat distributions over equally-likely states, if there are N physical states in the cloud at the start of the event, and M at the end of the event, then the entropy increase is given by $\Delta S = [\log (M/N)]$. For very small amounts of entropy increase $\Delta S << \log e = k$, this relation approaches $\Delta S \approx (M/N - 1)k$. Thus, if we want the entropy increase to be limited to, say, 0.01 k (for a quality factor of 4,000 given a 40 kT signal energy, or an energy efficiency of 99.975%), then this means that M can be at most ~1% greater than N.

Meanwhile, the reliability requirement means that at the end of a given operation, the cloud of physical states should be entirely or almost entirely confined within the set of "allowed" representations of a specific logical state. If not, then the logical state is in error and will have to be corrected. As we discussed earlier, logic errors represent a form of energy dissipation, and must be avoided in an efficient design. Conservatively, to attain an energy efficiency of f, the fraction of the cloud situated within the correct logical state should be at least f. In our example of .01 kT dissipation with a signal energy of 40 kT, the error probability should be no more than 0.025%.

Finally, of course our model's prediction of how the cloud of states develops must be physically accurate. The average entropy increase from any inaccuracy can be estimated from information theory as $\Delta S = \sum(a-p) \log p^{-1}$, where a is the actual probability of a given final state, p is the probability assigned to it in our model, and the sum ranges over all final states having either $a \neq 0$ or $p \neq 0$. (Note the average entropy increase may be unboundedly large if our model assigns 0 probability to any states that may actually occur.)

Despite the above stringent requirements on the accuracy of our model of the device's operations, in several ways the modeling and engineering requirements here are still much easier than in, say, full coherent quantum computing [9]. For one thing, our model does not need to predict exactly *which* initial states of the device will be taken to which final states; rather, we only need to characterize the set of probable final states. We also do not need to keep track of coherent quantum phase information; the initial and final states may both be highly decoherent, *e.g.*, they may be represented by near-diagonal density matrices, which are effectively just probability distributions over the system's set of natural "pointer" eigenstates.

5.2. Energy recovery mechanisms

It is important to realize that in reversible design, not only the flow of information through the machine, but also the flow of energy must be carefully tracked. A given chunk of energy of magnitude E (in excess of what a given system's energy would be in its ground state, that is, at zero temperature) is always carrying out transitions between distinguishable quantum states at the rate $2E/h$, unless the energy happens to be trapped within a cyclic process that is cycling among some small number N of states, in which case the transitions will proceed at the slightly faster rate $2(N-1)h/N$ [10]. Even the energy of a stable quantum ground state itself (relative to some lower reference level) is still always "busy" rotating the quantum phase of the system, at the angular frequency $\omega = E/h$.

So, energy is, by definition, always actively doing something. Thus, whenever we are finished using a given chunk of energy to carry out a desired logic transition, we must then immediately find some other job for that energy to do. For example, we might immediately redirect it into performing the next logic operation in a sequence. If there is not enough computational work available to keep the available signal energies occupied at a given point in a computation, then if we are smart, we

will arrange for the energy to then keep itself busy just "stirring" some part of the system's state that is already at its maximum entropy, since such activity won't increase entropy further. Or, we could arrange for the energy to participate in some more structured cyclic process, or to be locked into the ground state energy of some quasi-stable, newly-constructed state, which is what happens in capacitive and chemical energy storage systems.

If we don't find something useful or at least harmless for the energy to do, and we don't take pains to store it away, then the energy will busily occupy itself in dissipating out into the form of heat in the machine and/or its environment, resulting in entropy increase and a permanent consumption of free energy. Thus it is imperative to carefully design one's switching mechanisms so that we always carefully track where the signal energies go at all times, and make sure they are always either redirected into other useful purposes, or are carefully shepherded into some safe temporary storage facility. Whatever our choice, our design must carefully prearrange for it to occur automatically in the normal course of events, as the system propagates along through its natural sequence of states, under its own generalized inertia.

To do this is perhaps even more difficult than it sounds initially, since we not only have to store the energy away somehow when it is not being used, but we must also ensure that we don't lose track of its state (have an "expanding cloud" of possible states) in the meantime. The precise arrangement of energy among the various possible subsystems of the machine is part and parcel of our knowledge of the machine's state. If the arrangement of some energy ever becomes more uncertain than it was previously, in any way whatsoever, then this implies an increase of entropy, and the effective conversion of a portion of the energy to heat.

Most research on reversible energy storage mechanisms to date has focused on the design of resonant oscillating "power-clock" supply subsystems (*e.g.*, [11]), which are intended to keep the available energy occupied in a cyclic process of oscillation in the state of the power-clock resonator system. On each cycle, some of the energy of the resonator is borrowed to carry out a desired logical transition, where it is typically locked into place for temporary signal storage until some later cycle, when the transition is undone, and the borrowed energy is returned to the resonator cycle.

We must be careful in such processes to avoid leakage of information about the logical state of the machine into the oscillator subsystem, where it may pollute and corrupt the nominally "clean" oscillator signal, resulting in an increase in the entropy of this signal, and a significant dissipation of energy. In order for a signal to be recirculated around a cyclic path with a high quality factor (low rate of energy dissipation), the signal must have a low entropy bandwidth per quantum channel in the recirculation path. If entropy is being injected into the oscillator signal from the logic, then this will no longer be the case.

The potential alternatives to resonant oscillators for energy recovery, such as direct steering of signal energies into subsequent logic transitions, or the storage of signal energies in temporary quasi-ground states, have perhaps not yet been adequately explored.

5.3. Logical state encodings

A crucial question in the design of any logic device technology is: What will be the physical encoding of the logical states of the device? Of course, the answer in most conventional logic is that bits are represented by voltage levels (within some noise margins) that are stored on circuit nodes (wires and other capacitive elements). Reversible logic schemes using this encoding also exist (*e.g.*, see [12]).

However, we would be wise to also consider alternatives. For example, many quantum computing schemes use quantum spins (of electrons or atoms) to store information; a spin is a natural two-state system, so it is a convenient way to encode a bit. In superconductive circuits, information can be stored in the many forms, including the amount or direction of current flow in a loop, the phase of a macro-state wave function, or by current pulses propagating near-ballistically down superconducting transmission lines.

Some researchers have explored using the mechanical configuration of nanoscale solid or molecular structures to encode information. In optical computing, we can also encode information in electromagnetic waves or cavity oscillations, and in DNA computing, we encode information in the chemical composition of a (fairly dilute) solution of complex biomolecules.

Regardless of the physical medium, there are some general requirements that all information encoding schemes must satisfy.

First, the devices must maintain their logical state for long periods, at least, periods that are long compared with the time between accesses to the stored information. There are three basic ways to do this:

(1) Provide an energy barrier in configuration space that surrounds the set of physical states encoding a given logical state, so that in order for

the system to leave the logical state, it must first cross the energy barrier. Accidentally crossing the barrier would require either quantum tunneling, whose rate can be exponentially suppressed by making the barrier higher or wider, or else some form of thermal excitation, which can be exponentially suppressed by either making the barrier higher, or by lowering the temperature of the region near the barrier. With the barrier scheme, changing the logical state reversibly requires that at some point we must lower the barrier along some path between the old state and the desired new one.

(2) Don't use an energy barrier, but arrange for the spontaneous transition out of the logical state to require that the system traverse a large expanse of configuration space that is unlikely to be crossed by accident quickly. (This is like trapping someone in a forest by blindfolding them so that they will remain disoriented, and wander about at random.) A random walk takes expected time $\Theta(r^2)$ to proceed a radius r from its starting point. We can help our situation further if leaving the logical state requires passing through a narrow valley that is unlikely to be encountered. In the random-walk scheme, intended transitions between logical states are implemented by biasing the random walk in a direction that will lead to the desired state. (In a sense, an energy barrier can be considered a special case of this scheme, in which the narrowness of the "passage" is provided automatically by the rarity of states that have energy fluctuations high enough to get over the barrier. Similarly, the narrow passage can be viewed as having a low entropy, and thus a high free energy.)

(3) Don't have an energy barrier between states, or a barren expanse of intermediate states, but simply cool the system and isolate it from interactions with its environment so effectively that there is nothing that would cause it to change state. This is the approach used in spin-based quantum computing, in which the 0 and 1 states are right "next to each other," so to speak, with no energy barrier between them, but the states are stable unless perturbed, and interactions between the spin and the outside world that might perturb the state are suppressed. (Actually, it is possible to still view this as a form of the energy-barrier scheme; the path leading over the barrier is one where a large random thermal excitation from the outside world hits our system and causes it to change.)

Whichever of the three methods is used, we must ensure that the stored bits are reliable, in the sense that the probability of the system's spontaneously changing the stored logical state to an incorrect value should be small. As we discussed earlier, this implies that the energy difference (or more generally, the free-energy difference) between correct states and incorrect states (or paths leading to incorrect states) should be substantial compared to kT, and thus, in order for overall dissipation during logical transitions to be small compared to kT, energy efficiency must be very high.

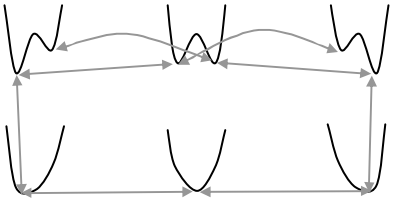

Figure 2. Adiabatic transitions between logical states separated by energy barriers.

5.4. Logical transition processes

How can we carry out highly energy-efficient transitions between logical states? Doing this requires that we use high-quality *adiabatic* or *ballistic* physical processes, which are processes in which only a small fraction of the energy involved in carrying out the transition is dissipated during the process. The distinction between adiabatic and ballistic processes is primarily just a difference in our analytical scope: in characterizing a given process as *adiabatic*, we are focusing on the energy dissipation within some small part of a system as it changes under the influence of external forces, whereas in characterizing a process as *ballistic*, we are focusing on the dissipation of an entire, well-isolated system as it proceeds through its natural evolution more or less autonomously.

Generally, any process that is overall ballistic involves adiabatic interaction processes between its subsystems. However, a system that includes adiabatic interactions between some of its subsystems may not be ballistic overall.

Given that what we care about in practice is usually total system energy dissipation, and not just the dissipation within an isolated subsystem, it must generally be the case for energy-efficient computing that the system as a whole proceeds forwards ballistically along the desired computational trajectory. However, in analyzing the design of such a system, it may be useful to pick out specific subsystems, and analyze them in terms of their adiabatic behavior under perturbations provided by neighboring ballistically-evolving subsystems.

For the case of logical states that are maintained reliably by the presence of potential-energy barriers (or more generally, free-energy barriers) separating correct physical states from incorrect ones, we can adiabatically change the logical state by

appropriately manipulating the energy surface, as in Figure 2. The gray arrows in the figure show adiabatic transitions between logical states that are located at stable or meta-stable local energy minima. We can symbolically represent the various logical states embedded in energy surfaces, and transitions between them in the following notation, which follows the cycle of states counter-clockwise from the high-energy state in the upper-right region of the figure: ◝ ↔ ◡ ↔ ◜ ↔ ◞ ↔ ◟ ↔ ◟· ↔ ◡· ↔ ◝·.

Let us trace through this sequence verbally. Suppose we are given an initial logical state that, at first, has energy that is greater (◝) than or equal to (◡) the desired new state, located to the right of the current state, which is denoted with a dot. We start by arranging for the energy of the initial state to be biased to a point lower than the desired new state (◜). Then we lower the barrier between the two states; now there is just a monotonic energy slope between them (◞). We now gradually slide the minimum-energy point to the right (◞ → ◟ → ◟·), and the logical state follows (while always remaining at a local energy minimum) until the logical state is in the correct configuration. Then, the barrier is raised (◝·), and, if desired, the state's energy can be boosted to some new default level, which may be either equal to (◡·) or greater than (◝·) the energy of old state. Then, the process can be repeated along some other path joining the new logical state to the next one after it in the desired computational sequence. Every step in this process merely involves energy transfer between the device that is storing the logical state in question, and some other device that is raising and lowering the energies at various points in the energy surface; the energy transferred need not be dissipated. Every step in this process, if performed slowly relative to the maximum transition speeds in the system, is guaranteed by the quantum *adiabatic theorem* to have an energy efficiency that can be made as large as desired.

Logical transition processes that can *never* be adiabatic, and that must therefore be mostly avoided in an energy-efficient design, include: (1) The lowering of an energy barrier that is preventing a higher-energy state from directly falling to a lower energy one, *i.e.*, the processes ◝ → ◟ and ◝· → ◞. This dissipates an amount of energy equal to the energy difference between the two states. Also forbidden is (2) the lowering of an energy barrier that distinguishes a known state from its neighbor when the two states have equal energy, *i.e.*, ◡· → ◟ or ◡· → ◟. Such a process loses 1 bit's worth of known logical information, and thus converts that bit into the form of k ln 2 amount of new entropy in the environment.

Conventional logic gates and memory cells perform the forbidden process (1) ubiquitously, for example, by turning on a transistor, which removes the potential energy barrier that prevents electrons on a high-voltage node from falling to a low-voltage node on the other side.

In adiabatic logic using transistors, we must never turn on a transistor when there is a significant voltage difference between its source and drain terminals.

General-purpose pipelined, sequential logic is still possible despite this constraint, and circuits as complex as complete microprocessors have been designed using this style. However, transistors do not have a particularly small value of their adiabatic *energy coefficient* or *energy-time constant*: this is the product $c_{Et} = E_{diss}t_{tr}$ of the energy dissipated E_{diss} in a given logic transition, and the time t_{tr} taken to perform that transition. For the adiabatic transfer of charge Q through resistance R, we have an energy coefficient of $c_{Et} = Q^2R$.

The energy-time constant is a particularly important figure of demerit for a device technology. The smaller the value of this constant is for a given device, the smaller is the number of those devices that will required to achieve a given level of performance within fixed power dissipation constraints. Intuitively, this is because a smaller value of this constant means that each device can run faster at a given level of power dissipation.

Analytically, given that $c = Et$ (suppressing the subscripts), and that the performance of the device (transitions per unit time) is $G = 1/t$, and that the device's power dissipation is $P = E/t$, solving for G in terms of c and P yields $G = (P/c)^{1/2}$. Thus, as the adiabatic energy coefficient decreases, and the allowable power P per-device remains fixed, the performance G of the device increases with $c^{-1/2}$.

Alternatively, if we have system-level power and performance requirements $P_{tot} = nP$ and $G_{tot} = nG$, in a system consisting of n devices of energy coefficient c, then solving for n gives the relation $n = cG_{tot}^2/P_{tot}$, or in other words, the number of devices required to simultaneously meet system-level power and performance constraints goes up linearly with c, and *quadratically* with performance! So, to scale up system-level performance by a given factor given a *fixed* number of devices n and a fixed power constraint P_{tot} requires a quadratically reduced value of c, given by nP_{tot}/G_{tot}^2. So, a crucial requirement for energy-efficient computing is to find alternative devices with a very low energy coefficient c, which requires low resistance in the

case of adiabatic transfer of a fixed charge. Superconductive devices seem very promising.

5.5. Synchronization mechanisms

Any computing scheme requires some mechanism for synchronization, to ensure that the inputs to some step in the calculation are available before the calculation is performed. The need for energy efficiency turns out to impose stringent requirements on synchronization mechanisms.

Essentially, the machine must be precisely engineered so that the delays along every calculational path are accurately known and predictable, and the expected arrival time of those signals must be built into the design of the components receiving those signals. This is because the accumulation of uncertainty in timing information is itself a form of entropy, and so it must be avoided in an energetically efficient machine.

Thermodynamically reversible asynchronous (self-timed) logic is likely to be impossible. Consider an element that is designed to wait until two input signals have both arrived, and then perform an operation. The information contained in the arrival time of the first input has nowhere to go, since the device is not supposed to respond at that time yet! The timing information contained in the first signal can therefore only be dissipated as entropy.

Thus, energy-efficient machines must be carefully, thoroughly clocked and controlled, and delays along all signal paths must be carefully matched. The normal scheme of driving adiabatic transistions using periodic oscillations of a power-clock resonator is one way to ensure that all logic remains synchronous.

5.6. Logic gates

A reversible logic gate is a device that can perform a logically reversible operation on the logical state of the subsystems accessed by that gate, and that (for energy efficiency) can do this in a nearly thermodynamically reversible fashion. As we said earlier, a logically reversible operation carries the set of possible "before" states one-to-one onto the set of possible "after" states.

Contrary to widespread misleading claims in the reversible computing literature, it is *not* necessary that (1) the gate hardware can perform only one operation, (2) we divide the subsystems accessed by the gate into fixed "input" and "output" signals, (3) we require the number of output signals to be equal to the number of input signals, (4) that the input signals must be consumed, (5) that the truth table of a gate with n input bits contain an output column that is a permutation of all 2^n input cases. In fact, all of these conditions only apply to special restricted families of reversible gates.

More generally, a reversible logic gate (1) can perform any of several operations, depending on the control signals applied to it, (2) it can access signals differently in the course of carrying out different operations (sometimes it can ignore a signal and use it as *neither* an input or an output, and sometimes it can use a signal as *both* an input and an output), (3) the number of signals modified (outputs) does *not* have to have any particular relationship to the number of signals measured (inputs), (4) the information present in the input signals need not be consumed, but may remain present on the input feeds after the operation is completed, and (5) the operation need not permute 1-to-1 *all* "before" cases, but only the *subset* consisting of those that will actually arise in the context of a particular design. Also, "gates" may, in general, contain some internal state information.

Thus, the design of reversible logic gates is actually much less tightly constrained than most of the reversible computing literature would have you believe. As a result, the literature is replete with sub-optimal hardware designs based on people taking the traditionally stated constraints too seriously, and failing to "think outside the box" and figure out the true requirement for reversibility.

The true requirement that must be imposed on the logical functionality of a reversible logic gate is simply this: that for each distinct operation that the gate can be directed to perform, no two initial logical states (local states of the device and the signals it accesses) that can possibly arise in the normal course of the machine's operation (given its design) can be transformed to the same final state.

Here are some examples of reversible logic operations that you won't find described in most reversible computing literature, but that are actually some of the easiest reversible logic operations to implement in practice:

rSET(x) – Reversible SET. Given the precondition that signal x is initially logic 0, change it to 1. This is conditionally reversible (on the condition that the precondition is satisfied). It can be implemented very easily in a way that is thermodynamically reversible when this condition is met. For example, in the energy-barrier picture, a control sequence that carries out the steps ∪ → ⁄ → ∪ → ∩ implements rSET; a gate supporting rSET can be implemented with just two transistors in standard CMOS technology.

rCLR(x) – Reversible CLEAR. Given the precondition that signal x is initially 1, change it to 0. Conditionally reversible. Can be implemented as

simply the time-reversal of rSET. One can build reversible storage elements out of gates that support just rSET and rCLR operations.

crSET(c, d) – Conditional reversible SET. Given the prediction that the initial state is not c = d = 1; if c = 1, perform rSET(d), else leave d unchanged. Conditionally reversible. crSET can be implemented in adiabatic CMOS using just two transistors, by a manipulation that performs either the sequence ↶→↺→↰ (d unchanged) or ↙→↓ →↘ (d taken from 0→1) on the output node d, conditionally on the input signal (which sets the barrier height).

crCLR(c, d) – Conditional reversible CLEAR. Given the prediction that the initial state is not c = 1, d = 0; if c = 1, perform rCLR(d), else, leave d unchanged. Conditionally reversible. Can be implemented in a way that mirrors crSET.

Gates that can perform just crSET and crCLR operations are easy to implement in CMOS and in many other technologies, and they are universal; they can be composed together into networks that reversibly implement any arbitrary combinational and sequential reversible logic! (Unfortunately, nearly all of the reversible computing literature ignores this very useful fact.)

rCOPY(s, d) – Reversible COPY. Given that initially d = 0, set d := s. Can be implemented in CMOS using four transistors, essentially by operating a crSET and crCLR in parallel, with the crSET controlled by s and crCLR controlled by ¬s.

un-rCOPY(x,y) – Given that initially y = x, set y = 0. Can be implemented as the time-reversal of rCOPY within the same gate hardware.

rMOVE(x,y) – Given that y = 0, set y = x and set x = 0. Can be implemented by the operation sequence rCOPY(x,y), un-rCOPY(y, x).

SWAP(x,y) – Swap x and y. Can be implemented given an internal signal t that is initially 0, by the operation sequence rMOVE(y,t), rMOVE(x,y), rMOVE(t,x).

Here are the reversible operations that are most frequently seen in the reversible computing literature, although they are usually misnamed as "gates." But they are really just operations.

NOT(x) – Toggle the logic value of signal x. This is often conceived as a gate having two signals x and y, where x is the input, y is the output, x is consumed, and y is produced. However, that is a more complex conception than necessary. The operation performed by such a gate would be more fully described as CONSUMING-NOT(x,y), which (on the precondition that y is initially a null value), sets y to the logical inverse of x, and then nulls x. But, the NOT(x) operation itself, strictly speaking, just flips the value of a single signal in-place.

Notice the above NOT(x) operation is distinct from an ordinary NOT gate or inverter, as it exists in conventional hardware today. The operation of an inverter would be best described as INVERT(x, y), which irreversibly overwrites y with the logical inverse of x, while leaving x itself unchanged. This operation is logically irreversible. Often you see the statement, "NOT is logically reversible," but this is misleading, since normally today NOT is implemented using the INVERT(x,y) operation, which is logically irreversible in its semantics.

cNOT(x, y) – Conditional NOT. Set y := x⊕y, where ⊕ denotes the Boolean exclusive-OR operator. Again, this is often misconstrued to mean that there must actually be *four* signals x_{in}, y_{in}, x_{out}, y_{out}, with the semantics being that x_{in} and y_{in} are consumed and x_{out} and y_{out} are produced. But again, this is more complex than necessary; instead a single signal y can be manipulated in place and used for both input and output, while x is a single signal used for input which is unmodified. Unfortunately, cNOT is not simple to implement in most technologies, so it is actually not a very good primitive operation for reversible logic in practice.

ccNOT(x, y, z) – Set z := xy⊕z, where xy is the Boolean AND of x and y. This is also called the "Toffoli gate" after its inventor [13]. Like cNOT, it is not easy to implement. However, it is widely discussed, since it is tied for simplest universal gate that is *fully* (as opposed to conditionally) reversible. (However, crSET and crCLR together are universal, they are easy to implement, and the conditionality of their reversibility does not preclude us from building thermodynamically reversible designs out of them.) Like NOT and cNOT, ccNOT is frequently misconstrued to require 3 consumed input signals and 3 separate output signals, when all that is really required are 3 total signals, where z is an in-out signal that is modified in-place, and the others are used as inputs only.

cSWAP(x,y,z) – Conditional SWAP. Swap y and z, if x = 1. This is also called the Fredkin gate after its inventor [13]. It is universal. Like the previous operations, in can be implemented by manipulation of two signals conditioned on a third, rather than by consuming 3 input signals and producing a separate output signal.

Multi-valued logics. Some (but not all) reversible logic schemes utilize three logic values, namely, 0, 1, and a third value "N" meaning "null" or "no information." The N state turns out to be useful because of the behavior of CMOS transistors. These schemes actually provide some of the simplest known transistor-based implementations of reversible logic [12]. For example, in 3-valued

logic, the following operations can both be implemented in a single gate using just two transistors:

rINVERT(x,y) – Reversibly invert. Given the precondition that y is initially N, set $y := \neg x$. Conditionally reversible. Implemented similarly to a simultaneous crSET and crCLR.

un-rINVERT(x,y) – Undo reversible inversion. Given the precondition that $y = \neg x$, set $y :=$ N. Implemented as the time-reversal of rINVERT(x, y).

And, using only five transistors, we can construct a gate that implements both:

rAND(x,y,z) – Reversible AND. Given the precondition that $z =$ N, set $z := xy$. Conditionally reversible.

un-rAND(x,y,z) – Undo reversible AND. Given the precondition that $z = xy$, set $z :=$ N. Conditionally reversible.

rAND and de-rAND, together with some simple latches, are universal for reversible logic. And, we can similarly implement 5-transistor reversible OR and un-OR.

Logic design using gates such as rINVERT, rAND, rOR, and their time-reversed versions is very similar to design using conventional logic, and is fairly efficient in terms of the number of transistors required.

5.7. Functional unit designs

Once one has a good library of primitive reversible logic gates, the design of higher-level functional units (such as registers, adders, decoders, multiplexers, etc.) is fairly straightforward and not too dissimilar from conventional logic design. However, we must keep in mind that we cannot just erase or overwrite information without suffering a thermodynamic penalty; rather, we must design our hardware algorithms to always just reversibly transform information in-place, using only the reversible operations such as rSET, rCLR, rCOPY and un-rCOPY, SWAP, NOT, rAND and un-rAND, cNOT, ccNOT, cSWAP, *etc.* For operations that are only conditionally reversible, we must be sure that their preconditions are met.

Generally, some complexity overhead is required in reversible designs in order to avoid information erasure. The usual strategy is use some initial information A to compute some needed intermediate information B, use B for some required purpose (such as computing further results C), and then "un-compute" or "de-compute" the intermediate information B in order to free up the space that it occupies, once it is no longer needed, so it can be reused to hold some new data. In order to decompute B reversibly, we have to be able to reconstruct what it is. One way that we can always do this, if the data A is still around, is to simply perform the time-reverse of the sequence of reversible steps by which B was computed from A to begin with. However, this approach requires us to keep A around until we are finished decomputing B, or to re-compute A from some previous information when it is needed to decompute B.

Another way to decompute B that is often more efficient when it is possible is to decompute B based on the information C that was subsequently derived from it. For this to work, C has to imply complete knowledge of B, that is, the Boolean function f that was used to compute C from B originally, $C = f(\mathrm{B})$, must be an invertible function. If it is, then we can reconstruct B using $\mathrm{B} = f^{-1}(C)$, and thus we can decompute B by performing the time-reversal of this means of calculating B. The advantage of this approach is that A can be decomputed before B is and does not need to be recomputed; this can end up saving us a lot of space and/or time in long computations.

However, this approach is sometimes intractable even when it is possible in principle. For example, suppose B consists of a pair of large prime numbers p, q with $p \leq q$, and suppose $\mathrm{C} = f(\mathrm{B}) = f(p, q) = pq$, the product of p and q. In principle, p and q are uniquely determined by their product pq, and so the function f is actually invertible (given the constraint that $p \leq q$). However, it is not known to be tractable to invert this particular function f, unless we have a quantum computer, since there is no known fast classical algorithm for factoring. Therefore, although decomputing B from C is possible in principle in this case, it cannot be done efficiently without a quantum computer (as far as we know). We note f is an example of what is known as a *trapdoor* or *one-way* function, an easy-to-compute function that, although it is invertible in principle, is not known to have an efficient algorithm for computing its inverse. Although it has never been proven that any one-way functions actually exist, many functions (such as the f above) are widely conjectured to be one-way.

The problem of iterating a one-way function is conjectured to be a problem for which reversible algorithms have strictly greater spacetime complexity (number of "device-cycles" occupied) than irreversible algorithms [14].

Because of the complexity overheads of reversible logic in cases such as this, the most cost-efficient hardware designs in practice in many cases will be hardware that is not *fully* reversible, but only reversible to a limited extent, determined by the level of energy efficiency we are trying to achieve. In general, we must perform a systems-engineering optimization over the design parameters (such as

was done in [15]) in order to find the solution that is overall most cost-efficient. However, as energy coefficients and device costs decrease, the optimal designs will be reversible to greater and greater degrees (approaching ever closer to 100%). This is true *despite* the algorithmic overheads of highly reversible design.

5.8. Processor architectures

In the near term, there will be little advantage to be gained from applying reversible logic above the level of the design of small functional units. But, as our desired level of energy efficiency becomes ever closer to 100% (for ever-greater performance within fixed power constraints), designs will have to be made reversible throughout larger and larger blocks of computational work, up to the level of even complete CPU cores. Eventually, there will even be a need for the programmable architecture itself to reflect the underlying reversibility of the logic. This is because the algorithmic overheads of reversible computing imply that we cannot expect the machine or the compiler to automatically find a good reversible algorithm for implementing a computation that the programmer has specified in a non-reversible fashion. In general, the most efficient reversible algorithm for performing a given task may be structured very differently from the most efficient irreversible algorithm for performing the same task [6].

Thus, microprocessor and DSP instruction set architectures and FPGA architectures will eventually need to change to allow the programmer to specify his software and hardware algorithms directly in terms of underlying reversible primitives. A number of designs for reversible instruction-set architectures already exist (*e.g.*, 16), which include instruction-set level analogs of reversible logic gates, and special reversible branch instructions to enable reversible control flow. Other than that, these instruction sets are fairly ordinary. Many common instructions (such as one-operand NEG and two-operand ADD) are already logically reversible. Others can be replaced with reversible variants.

5.9. Design languages

By the phrase "design languages" I mean to include both hardware description languages (such as VHDL and Verilog) and high-level software programming languages (such as C and Fortran), since both types of languages can be used to describe Turing-complete computational algorithms.

For the reasons described in the previous section, design languages will eventually need to adapt to allow programmers to craft reversible algorithms directly in terms of reversible language primitives. Purely automatic substitution of reversible constructs for irreversible ones will in general lead to sub-optimal efficiency. A few high-level reversible programming languages exist (see [6]), and I have recently begun working on the design of a reversible hardware description language. However, these languages are not yet very sophisticated, and as well their development is somewhat premature, given that we are still quite a long way away from having a high-quality reversible device technology that would create significant demand for such languages.

For the most part, reversible languages can look fairly conventional, but there are some differences:

(1.) Assignment to variables is deprecated, in favor of binding and operations like +=.

(2.) Control-flow constructs are time-symmetric.

(3.) Subroutines can be called in reverse!

(4.) Automatic garbage collection is deprecated, because it necessarily creates entropy.

5.10. Application algorithms

Finally, once the need for something like 99.9999999% energy efficiency has (probably many decades from now) forced us to migrate to "reversibility-aware" computer architectures and design languages, we (that is, programmers and special-purpose hardware designers, as well as mathematical algorithmicists) will have to get accustomed to crafting our high-level application algorithms (or at least, their most power-hungry portions) in terms of the reversible paradigm. Reversible algorithm design is not particularly difficult, and one can always fall back on applying known general transforms that map arbitrary irreversible algorithms to equivalent reversible ones. But reversible design will be initially unfamiliar to most computer engineers and programmers, who, ever since the days of Ada Lovelace, have trained themselves to think in terms of primitive operations that are logically irreversible. However, I am confident that as soon as reversible design actually becomes demonstrably useful for dramatically improving computer energy efficiency and application performance, designers (at least of high-end applications) will rapidly proceed to learn and adopt it.

6. Conclusion

Computer performance, at realistic levels of power consumption, is fundamentally limited by the energy efficiency of the low-level operations within the machine. The magnitude of signal energies is soon reaching firm thermodynamic limits, but what is not so widely recognized is that the magnitude of signal energy itself does not mean that this energy must necessarily be dissipated when performing operations. Instead, we can in principle recover and reuse a fraction of the signal energy that, as far as we know, can be made to approach 100% as closely as we like, given sufficiently aggressive engineering. This, in turn, can theoretically enable computers to run as fast as we like (in terms of overall parallel performance) at a given level of power dissipation, with total performance being fundamentally limited only by the total amount of energy that we invest in our computing mechanisms. As far as we know, approaching the physical limits of computing ultimately means just (1) harnessing larger and larger total quantities of energy in the service of computing, and (2) using that energy with greater and greater efficiency, so that waste heat still remains manageable.

However, we have seen in this document that increasing the energy efficiency of computers to values close to 100% is actually quite a challenging proposition, requiring us to break new ground in high-precision engineering of the machine at every level, from how we model its nanoscale physics to the way we organize and clock the logic, and to how we structure our algorithms. As of today, electrical engineers have not even yet attained the level of energy efficiency that would be needed to circumvent the eV-scale reliability barrier on signal energies and bring dissipation close to the room-temperature kT level where reversible logic would be required for further progress. And, sub-kT computing with reversible logic itself requires incredibly precise modeling, manufacturing, characterization, and control of devices and clocking systems, long before we get to the point where reversible logic design and programming starts to begin to become very helpful.

So given all this, why should we study reversible computing? Why not instead just give up on it, and resign ourselves to the conventional logic paradigm, and just tackle the conventional goal of reducing dissipation per op (and increasing performance-per-power) by a factor of 10,000 or so, from today's ~fJ/op levels down to the neighborhood of 1 eV? In principle, none of the complex set of requirements discussed in the previous section are required to reach this level.

Well, this is a good point, but there are three main reasons why I think that we should *at least* not *forget* about reversible computing, and ideally devote a reasonable amount of resources to its study even today, enough to keep the field alive.

(1) Research into reversible computing, if it yields demonstrated successful systems, offers the possibility of jumping ahead of the technology curve that we are on today, and achieving higher levels of performance even sooner than the 20-year timeline on which conventional technology must stall.

(2) Twenty years is, after all, not so far away, and we will find ourselves there before we know it. If, in the meantime, reversible computing research hasn't been pursued aggressively, then solutions won't be available and computer performance will definitely stall. This could even happen much sooner than we think, in 10 years or perhaps sooner, if technical difficulties in scaling the conventional approach prevent us from getting all the way down to the 1 eV/op dissipation level. If computer technology stalls, and we become used to a state of stagnation rather than progress, then it may take a long time to restart the engine of technological innovation. In contrast, if reversible technologies are ready to be put into place by around 20 years from now, then the trend of progress in computer performance will experience barely a bump, and will continue indefinitely far onwards into the future. This would be a boon for the high-tech economy, and for the world overall.

(3) Finally, I think it is *really important* that we don't completely drop reversible computing and forget about it, however challenging and difficult it may be. In fact, remembering and pursuing reversible computing may be the *most important thing that we can possibly do*. This is because in the long run, it may literally have *infinite* value, in the sense that it may make the difference between a *finite* and an *infinite* future for our entire civilization, or more generally, for all life in the universe!

This last claim needs some elaboration. Noted astrophysicists Krass and Starkman [17] have argued that, even if our civilization someday colonizes distant stars, the total amount of energy that we can harvest in the universe (before the rest expands to be forever beyond our reach) is *finite*. Even the maximum entropy of the entire observable universe

itself is finite if (as present observations suggest) Einstein's cosmological constant has a fixed non-zero value. Therefore, if there is any fixed lower bound greater than zero on the energy dissipated or entropy generated by a computational operation, then our civilization (and all life) will necessarily eventually run out of free energy (entropy will reach a maximum) and all interesting activity will then cease. All civilization and life will only have performed a finite number of organized operations that could be construed as "thoughts" or "computations."

But, in contrast, suppose that Landauer and Bennett are correct, and there really is no lower limit on entropy generation per operation. Then, suppose that sometime before half of our available energy is used up, we figure out how to use the remaining half twice as efficiently as before (that is, with half the entropy increase per operation). Then, before half of the remaining half is used up, we figure out how to use the rest twice as efficiently again. And so on. With each half of the remaining energy, we accomplish an equal number of computational operations. Thus in principle, we (or our postulated machine "descendants") could perform literally an *infinite* number of computational operations (*i.e.*, have an "infinite number of thoughts") using only a finite supply of energy.

Reversible computing is, in fact, the one and only possible way to "save the universe" from doomsday scenarios like Krauss and Starkman's that is both (1) consistent with known fundamental physics, and (2) doesn't depend on the existence of improbable new hypothetical phenomena such as wormholes to other universes, *etc.* Fundamentally, this plan only depends on our ability to model the laws of physics ever more precisely over time, and to isolate subsystems from the unknown external environment more and more thoroughly, neither of which seems particularly implausible. But, this kind of plan can *only* work if we (1) achieve highly efficient reversible computing, and (2) make it more and more energy-efficient over time.

And further, to be a bit more down-to-earth, if one can judge by the price of gas these days, and by the declining rate of world oil production, we might face an energy shortage within the next few decades, and not necessarily only billions of years from now. As world oil supplies dwindle, it might be smart for us to have a viable research program aimed at continuing to improve our computational capabilities without suffering proportional increases in the rate at which we use up our limited supply of fossil fuels, and other finite energy sources.

Therefore, it just might be a very wise idea for us to seriously get started on reversible computing research today, and try to figure out exactly *how* to engineer the new nanodevices, clocking systems, and architectures that are required by this one and only possible way to circumvent the limits of the conventional digital logic paradigm, and produce computers that are ever more energy-efficient, so that we can approach the *true* fundamental limits of computing, however far away they may be.

For all we know today, the ultimate fundamental limits of computing may only be the cosmological ones that will be met in a distant future, billions of years hence, after we have (potentially) converted all visible galaxies into a single, giant 99.999(lots more 9's)% reversible supercomputer, running some sort of enormous virtual "Matrix" for our uploaded minds to work and play in virtually forever, until eventually we run out of new states to explore, new things to do, new thoughts to think.

Although this picture is only a fantasy today, we cannot know whether such a future is scientifically possible unless we work seriously to try to achieve it, and we aren't going to come anywhere close to achieving anything like this unless we seriously and aggressively pursue the technological possibilities that reversible computing offers us.

Which future will it be? Overheated computers, technological stagnation, and possible economic decline? Or, ever more energy-efficient, cool-running, ballistic reversible computers, leading to a literally unbounded possible future of growth and prosperity? The answer may depend on what fields we researchers choose to devote our skills and efforts to today.

I implore my audience: Study physics. Help invent new kinds of nanodevices with high adiabatic energy coefficients. Design, build, and empirically test high-quality ballistic oscillators, interacting with quasi-static logical states, driving adiabatic transitions between them. Systematically find and eliminate sources of dissipation in your prototypes, one by one. Extend your designs to larger and larger scales of complexity, with larger and larger logic blocks ever more tightly and precisely synchronized. Design fully-reversible architectures, languages, and algorithms.

It is only through intense efforts roughly along these lines that we can possibly avoid approaching firm limits on the raw, low-level energy efficiency of computers within our lifetimes. And, if we do not create reversible computing ourselves, there is no guarantee that someone else will do it for us. Since high-quality reversible computing is so difficult to achieve at the lowest level of devices and oscillators, many researchers who have studied the field in the past have abandoned it, to turn to easier pursuits. Many have tried to justify this choice by

dismissing reversible computing as impossible, but no logically valid justification for this impossibility claim has ever been offered. But if people continue giving up on reversible computing too easily, we'll never know if it can be done.

It is only by letting ourselves admit the physical possibility of reversible computing, while bravely and persistently tackling the difficult physics and engineering challenges associated with realizing it in practice, that we may hope to achieve significant progress in computer performance beyond the next decade or two. I urge all readers of this paper to take upon themselves some small part of the responsibility for helping to meet this great 21st-century engineering challenge, to open up grand, unbounded new vistas for the future of computing.

References

[1] Gordon E. Moore, "Cramming more components onto integrated circuits," *Electronics*, April 19, 1965, pp. 114-117.

[2] G. E. Moore, "Progress in digital integrated electronics," *Technical Digest 1975 International Electron Devices Meeting*, IEEE, 1975, pp. 11-13.

[3] D. J. Frank, "Power constrained CMOS scaling limits," *IBM J. Res. Dev.*, 46, 2/3, 2002, pp. 235-244.

[4] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Dev.*, 5, 1961, pp. 183-191.

[5] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Dev.*, 17, 6, 1973, pp. 525-532.

[6] M. P. Frank, *Reversibility for Efficient Computing*, Ph.D. thesis, Massachusetts Institute of Technology, 1999.

[7] J. L. Hennessy, D. A. Patterson, and D. Goldberg, *Computer Architecture: A Quantitative Approach*, 3rd ed., Morgan-Kaufmann, 2002.

[8] D. Deutsch, *The Fabric of Reality*, Penguin Books, 1998.

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge, 2000.

[10] N. H. Margolus and L. B. Levitin, "The maximum speed of dynamical evolution," *Physica D*, 120, 1998, pp. 188-195.

[11] V. Anantharam, M. He, K. Natarajan, H. Xie, and M. P. Frank, "Driving fully-adiabatic logic circuits using custom high-*Q* MEMS resonators," in *Proc. Int. Conf. on Embedded Systems & Applications*, CSREA, 2004, pp. 5-11.

[12] S. G. Younis and T. F. Knight, Jr., "Asymptotically zero energy split-level charge recovery logic," in *Int. Wkshp. on Low Power Design*, 1994, pp. 177-182.

[13] E. F. Fredkin and T. Toffoli, "Conservative logic," *Int. J. Theo. Phys.*, 21, 3/4, 1982, pp. 219-253.

[14] M. P. Frank and M. J. Ammer, "Relativized Separation of Reversible and Irreversible Space-Time Complexity Classes," manuscript submitted to *Information and Computation*, May 2001.

[15] M. P. Frank, "Nanocomputer systems engineering," in *Nanotech 2003: Tech. Proc. of the 2003 Nanotechnology Conf. and Trade Show*, vol. 2, Computational Publications, 2003, 182-185.

[16] C. Vieri, *Reversible Computer Engineering and Architecture*, Ph.D. thesis, Massachusetts Institute of Technology, 1999.

[17] L. Krauss and G. Starkman, "Life, The Universe and Nothing: Life and Death in an Ever Expanding Universe," *Astrophysical Journal*, 531, 2000, p. 22.